L14: Entry 6 of 7 File: TDBD Dec 1, 1985

TDB-ACC-NO: NN85123140

DISCLOSURE TITLE: FACTOR (Fairchild Sentry VIII Tester Language) SYNTAX Checker/Analyzer

PUBLICATION-DATA:
IBM Technical Disclosure Bulletin, December 1985, US

VOLUME NUMBER: 28
ISSUE NUMBER: 7
PAGE NUMBER: 3140 - 3144

PUBLICATION-DATE: December 1, 1985 (19851201)

CROSS REFERENCE: 0018-8689-28-7-3140

DISCLOSURE TEXT:

- By providing quick feedback of syntax errors of a FACTOR program on a system which is independent of the single user target computing system (Sentry VIII), a significant reduction in the amount of development time of test programs is realized. Since it allows multiple programs to be developed and syntax checked on an IBM system, the Sentry VIII system is free to do its primary job of testing integrated circuits. This article describes a grammar parser written to accept the FACTOR programming language developed by Schlumberger Fairchild. The parser has been implemented in PL/1 and 370 Assembler languages. It is a bottom-up parser which is based upon a grammar in "Backus-Naur Form" (BNF) and has 4 basic parts which allow the parsing algorithm to be accomplished. These are as follows: A. PARSER routine - this routine makes the decision of what to do with the present state given its previous input (held on the parse stack) and its current input token (provided by the scanner). The four basic decisions that can be made by the parser are: 1) SHIFT - place the current input "token" onto the top of the parse stack after moving all previous items on the stack down one position (the actual "stack" is implemented as an array, with a variable as the stack pointer; i.e., no movement of data within the array is required to place or remove items from the parse stack). 2) REDUCE - locate an appropriate rule that can be applied to the current contents of the parse stack and replace them with the result of the rule's application (these are defined by the BNF grammar). 3) TRIPLE - the decision must be made by using the 2nd most recent item placed on the parse stack in addition to the token at the top of the parse stack and the current input token provided by the SCANNER. 4) SYNTAX ERROR - if none of the above decisions is indicated by the top of the parse stack and the current input token, then a syntax error has occurred and an appropriate error message is displayed. Notes: 1. Parsers are generally referred to as "state machines" in Finite State Automata Theory. 2. EVERY SYMBOL USED WITHIN THE GRAMMAR IS ASSIGNED A unique numeric value. This number is referred to as a "token" and is identified by the SCANNER for use by the PARSER. For example, each keyword is assigned an individual "token" value, each symbol is assigned an individual value, and a given identifier is also assigned a unique token value. B. EMITTER routine - this routine processes the "actions" that need to be acted upon when a reduction is to be made of the parse stack. This is needed in order to generate a more complete and informative source listing for the user as well as to process special directives to the compiler, such as the PAGE, IFC, INSERT, and MACRO statements. C. SCANNER routine - this routine processes the input characters (provided by the GETCHAR routine) and forms basic "tokens" for processing by the parser. D. GETCHAR routine - this routine provides the primary input of characters to the SCANNER from the FACTOR source program being analyzed. It also provides the SCANNER with the classification or the type of character (whether it is numeric, alphabetic, special character, etc.) that is currently being processed. This information is used by the SCANNER to make the task of identifying the particular input token assignment simpler. This provides syntax analysis of FACTOR programs; a display of nesting levels of the block control structures BEGIN...END; and FUNCTION/SUBROUTINE...END; allows listing of statements excluded from compilation by appropriate conditional compile statements; detects duplicate variable names that are greater than 8 characters due to truncation by the FACTOR compiler; and provides summary listing of all errors detected throughout the program. This simplifies locating warning error messages which are buried within the listing when the Fairchild FACTOR Compiler is used (the compiler will only report the number of actual errors within a given program even when it flagged statements with a warning; thus the compiler may report no errors even when there are potentially erroneous statements present). The entire program is represented in the flow diagrams.